Independent Component Analysis - En jämförande laboration mellan fastICA, SNG samt AMUSE.

David Andersson

February 5, 2016

Abstract

Syftet med rapporten är att redogöra för olika metoder som används i independent component analysis (ICA). Vidare så används de olika algoritmerna på olika typer av data, nämnligen vit icke gaussisk, icke vit gaussiskt samt faltad.

1 Introduktion

Den första delen kommer att redogöra för de olika metoder som används. Efteråt så tar jag upp lite om vilka typer av datasets som används, och hur de olika relevanta algoritmerna presterar. Slutligen så kommer också en jämförelse av resultaten samt slutstats.

För att lösa problemen används Matlab. All teori kring rapporten kommer från boken Independent Component Analysis, skriven av Aapo Hyvärinen, Juha Karhunen och Erkki Oja [1].

Rapporten utgår ifrån en grundläggande förståelse för principle och independent component analysis.

2 Metod

2.1 Algoritmer

Gemensamt för algoritmerna är att deras uppgift är att separera ljudkällor med hjälp av observationer av ljudkällorna. Som en estimering för den högre ordningens statistik som behövs så används icke linjäriteter. Framförallt har tangens hyperbolicus använts. försök med x^3 förekomm också, men utan förbättrade resultat, vilket gjorde att jag inte tog med resultaten i rapporten.

2.1.1 fastICA

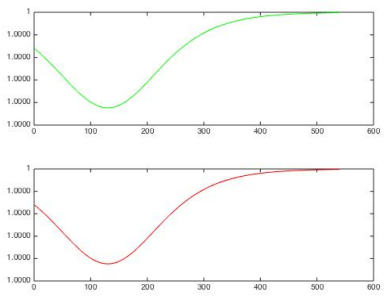

Figure 1: fastICA error function.

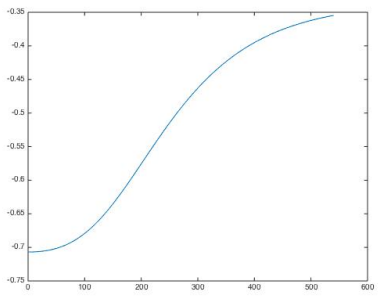

Figure 2: fastICA skalärprodukt.

FastICA bygger på att hitta en orthogonal rotation av datan (som är okorrelerad) så att datan blir oberoende. Detta görs genom att använda fixed point iterationer enligt $\{Xn + 1 = f(Xn), n = 0, 1, 2, ...\}$ i ett försök

att maximera icke-gaussiskhet. För att göra det så approximeras negentropy genom angivna ickelinjäriteter. På resultatet utförs en gradienten sökning för att fastställa vilken riktning algoritmen ska fortsätta. För att slippa error cumulation så avände jag mig av en symmetrisk ortogonalisering.

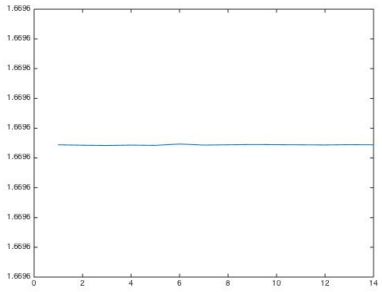

Figure 3: fastICA matrix condition.

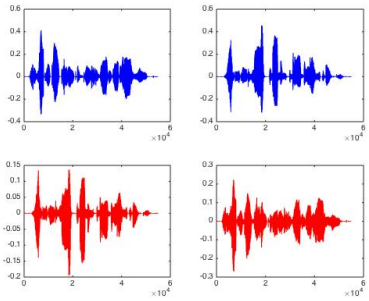

Figure 4: fastICA. Blå: Orginal, Röd:Resultat.

För att mäta konvergensen så skulle vi använda matlabs matrix condition för W*A, alltså relationen mellan det mista och största värdet, som borde vara 1. Detta fungerade inte särskilt bra i min implementering, så istället så provade jag två andra metoder: Mean square error (MSE) mellan inversen av A och W, samt skalärprodukten mellan Wn och $Wn + 1$. Detta kan ses i figur 1, 2, 3. En observation är att jag fick ungefär samma resultat med de två senare metoderna, men matrix condition är konstant. För att visa på att separtionen är utförd i någon mån så ses de olika signalerna samt resultatet i figur 4. För att ytterligare styrka konvergensen så kan man se en scatter plot i figur 5 Angånende konvergensen så kan man säga att om A var vald rätt (lyckligt slumpad) så var konvergensen snabb, men i många fall så konvergerade algoritmen aldrig eller mycket dåligt.

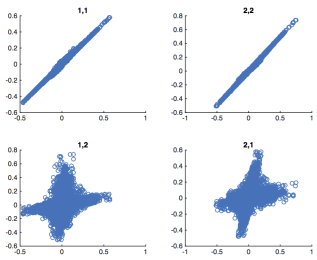

Figure 5: fastICA scatter plot.

2.1.2 SNG

Stochastic natural gradient (SNG) är likt fastICA en fixed point algorithm som använder sig av gradient och strävar efter att maximera eller minimera kurtosis (gaussiskhet). Detta görs genom att välja olika icke-linjäriteter. För denna labens skull så valde jag att enbart använda icke-linjäritet för supergaussiska distributioner då det stämmer bäst över med mänskligt tal. För att slippa error cumulation så avände jag mig av en symmetrisk ortogonalisering.

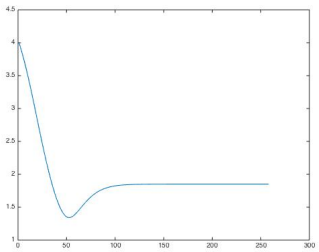

Figure 6: SNG matrix condition.

Likt fastICA så konvergerade matrix condition aldrig till 1, men jag fick ändå bra resultat när någon form av konvergens var uppnådd. För att se till att ha rätt steglängd så avnände jag mig av två loopar, den yttre minskade steglängden och den inre itererade algoritmen. Detta gjorde att konvergensen var resonabelt snabb och lyckades nästan alltid att åstakomma någon form av separation.

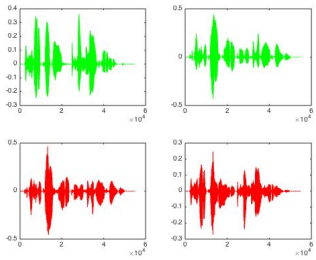

Figure 7: SNG Grön: Orginal, Röd: Resultat.

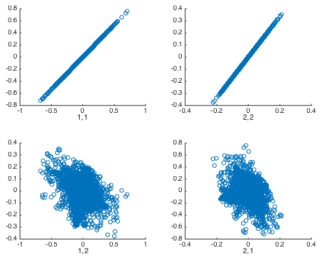

Figure 8: SNG scatter plot.

För att visa på konvergen så finns bilder på ett relativt typiskt försök. Figur 6 visar hur matrix condition brukar konvergera. Figur 8 visar en scatterplot mellan olika kombinationer av orginalsignalen och resultatet, vilket visar på linjära samband mellan orginal och resultat. Figur 7 visar orginalsignalen samt resultatet av SNG. Försöket slutade på steglängden 0.0417.

2.1.3 AMUSE

Till skillnad från de andra algoritmerna så bygger AMUSE på en tidsförskjutning (τ) av den vitade datan. Den vitade och förskjutna datan beräknas sedan som en kovariansfunktion enligt $E\{z(t) * z(t - tau)'\}$. För att det ska fungera så måste egenvektorerna vara unika. Detta sker enbart om den tidsförskjutna kovariansfunktionen är unik för alla komponenter, därför måste vi anpassa tau. AMUSE kräver alltså inte att signalerna är oberoende och är därför lämpade till gaussiska distributioner.

3 Resultat

3.1 Datasets

3.1.1 Laplaciska

Datasetet genereras enligt instruktioner i laben men medelvärde 0 och varians 2.

fastICA konvergerar efter 11 iterationer och resulterar i matriserna:

$$A^{-1} = \begin{bmatrix} 10.5942 & 8.3104 \\ 8.1448 & 5.6782 \end{bmatrix} \tag{1}$$

$$W = \begin{bmatrix} -0.8146 & -0.5800 \\ -0.5800 & -0.8146 \end{bmatrix} \tag{2}$$

$$cond(W * A) = 37.1397 \tag{3}$$

SNG konvergerar efter 18 iterationer med steglängd 0.0556.

$$A^{-1} = \begin{bmatrix} -2.7485 & 4.9379 \\ 1.4967 & -0.3794 \end{bmatrix} \tag{4}$$

$$W = \begin{bmatrix} -0.0567 & 0.0828 \\ 0.4127 & 0.0114 \end{bmatrix} \tag{5}$$

$$cond(W * A) = 19.5273 \tag{6}$$

3.1.2 Filtrerade

För att analysera fastCA och AMUSE angående deras prestation med filtrerade signaler (gaussisk och laplacisk) så användes en funktion E1 enligt handlendingen. I tabel 1 nedan ser vi max, min och medelvärde av funktionen över 1000 iterationer. Vad man kunde se var att fastICA presterar marginellt bättre i båda fallen, men under väldigt snäva konvergenskrav. Dock så borde AMUSE teoretiskt prestera bättre på gaussiska signaler. En anledning kan vara att jag enbart provade med $\tau = 1, 2$, enligt manualen. Algoritmen skulle också kunna utökas genom att ta hänsyn till flera τ åt gången. Enligt teorin så borde inte fastICA vara bättre än PCA för gaussiska distributioner, vilket jag också provade med resultatet min: 0.013, mean: 1.1461, max: 3.428.

Table 1: Jämförelse av fastICA och AMUSE

fastICA						AMUSE					
Gaussian			Laplacian			Gaussian			Laplacian		
min	mean	max	min	mean	max	min	mean	max	min	mean	max
0.083	1.771	3.696	0.075	1.762	3.806	0.318	2.036	3.827	0.256	2.025	3.860

3.1.3 Tal

Både fastICA och SNG konvergerade relativt snabbt för två signaler. Som tidigare så fungerade fastICA bättre när den väl konvergerade, men SNG konvergerade oftare. I figur 4 respektive 7 för fastICA respektive SNG finns resultatet från en lyckad separering.

3.1.4 Faltningar

Jag kunde höra en liten förbättring med fastICA, med väldigt snäva konvergenskriterier. Dock hörde jag ingen skillnad med AMUSE, det lät framförallt som en volymökning/sänkning.

3.1.5 Utökning till fler signaler

För 3, 4 och 6 signaler så blev proceduren genast mycket svårare. Jag fick sämre resultat och längre konvergeringstid. Kvalitéen på utsignalen för fastICA kunde hjälpas lite med hjälp av att sätta snävare konvergenskriterier. Dock så fann jag att matrix condition som jag använde till SNG inte längre var lika effektivt. Konvergensen var relativt snabb vid rätt steglängd, men jag fick sällan bra separation. För att få ett bättre resultat så använde jag mig av bokens tips att kolla när $E\{g(y) * y'\} = I$ Vilket visade sig vara ett mycket tacksamt mått. Som man kan se i figur 9 så hamnar algoritmen snabbt nära enhetematrisen, dock så kan det krävas många iterationer om toleransen $(I - E\{g(y) * y'\}) < tol$ är för snävt.

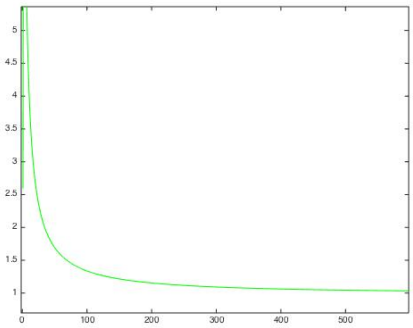

Figure 9: Resultat av nytt konvergenskriterie.

References

[1] Hannu Oja and Klaus Nordhausen. Independent component analysis. *Encyclopedia of Environmetrics*, 2001.